

Dynamic Programming, Reinforcement Learning, Message Passing

Sungjoo Ha

December 27th, 2017

Puzzle #1

주사위를 6이 나오기 전까지 계속해서 던진다면 평균적으로 몇 번 던져야 할까?

Approach #1

Intuition

직관에 의해 6

Approach #2

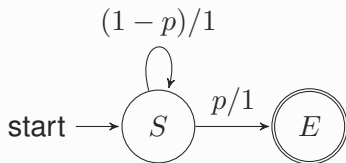
Infinite Series

한 번 주사위를 던졌을 때 6이 나올 확률이 p 라고 하면,

$$\begin{aligned}S &= p \cdot 1 + p(1-p) \cdot 2 + p(1-p)^2 \cdot 3 + \dots \\(1-p)S &= p(1-p) \cdot 1 + p(1-p)^2 \cdot 2 + \dots \\S - (1-p)S &= p(1 + (1-p) + (1-p)^2 + \dots) \\&= p \sum_{x=0}^{\infty} (1-p)^x \\&= p \cdot \frac{1}{1 - (1-p)} \\pS &= 1 \\S &= \frac{1}{p} \quad \square\end{aligned}$$

Approach #3

Markov Reward Process



Bellman equation을 사용하여 state S 의 value를 계산,

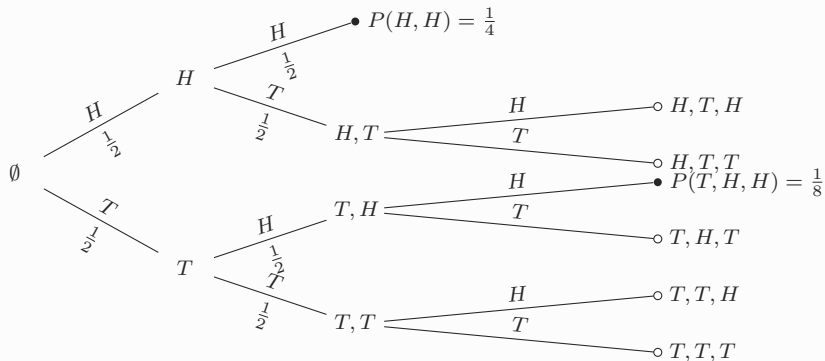
$$\begin{aligned}V(S) &= 1 + (pV(E) + (1-p)V(S)) \\ &= 1 + (1-p)V(S) \\ pV(S) &= 1 \\ V(S) &= \frac{1}{p} \quad \square\end{aligned}$$

Puzzle #2

동전을 계속 던졌을 때 두 동전이 연속으로 앞면이 나올 기대 시행 횟수는?

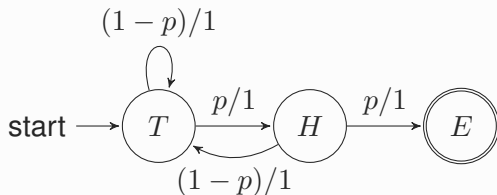
Approach #1

Probability Tree



Approach #2

Markov Reward Process



Bellman equation을 사용하여 state T 의 value를 계산,

$$\begin{aligned}V(T) &= 1 + ((1-p)V(T) + pV(H)) \\V(H) &= 1 + (1-p)V(T)\end{aligned}$$

연립방정식을 풀면, $V(T) = \frac{1+p}{p^2}$ □

RL Perspective

- ▶ Markov process model을 세운다.
 - State set
 - Action set
 - Transition probability
 - Reward distribution
- ▶ RL 접근 방법을 사용한다.
 - Solve Bellman equation by matrix inversion (MRP)
 - Value iteration

Puzzle #3

[0, 1] 균등분포에서 샘플링한 값을 1이 되기 전까지 계속 더한다면
평균적으로 몇 번 샘플링 해야하는가?

MRP Approach

- ▶ 총합이 1이 넘기까지 얼마나 남았는지를 상태로 삼는다.
- ▶ No action
- ▶ 다음 상태로 넘어갈 확률은 균등분포로부터 샘플링된다.
- ▶ Reward는 매 시행마다 1

$V(s)$ 는 앞으로 s 만큼의 합이 더 나와야 하는 상태의 가치이다. 시작 상태는 1이며 이 때의 가치 $V(1)$ 을 구하는 것이 목표이며 $V(0)$ 은 0이다. Bellman equation, $V(s) = 1 + \sum_{s'} P_{ss'} V(s')$ 을 풀도록 한다.

MRP Approach

$$\begin{aligned}V(s) &= 1 + \int_0^1 V(s-u)du \\ &= 1 + \int_0^s V(s-u)du\end{aligned}$$

Let $s - u = t$

$$\begin{aligned}V(s) &= 1 + \int_s^0 -V(t)dt \\ &= 1 + \int_0^s V(u)du\end{aligned}$$

MRP Approach

Differential equation을 세우면,

$$\frac{dV(s)}{du} = V(s)$$

미분해서 자기 자신이 되는 함수는 지수함수이므로,

$$V(s) = C \cdot e^s$$

Boundary condition에 의해,

$$\begin{aligned} V(0) &= 1 + \int_0^0 V(u)du \\ &= 1 \end{aligned}$$

이로부터 $C = 1$ 임을 알 수 있다. 그러므로 $V(1) = e$ □

Puzzle #4

52장의 플레이잉 카드를 잘 섞은 후 뒤집어 쌓아놓습니다. 매 턴마다 (1) 가장 윗 카드를 뽑아 뒤집거나 (2) 그대로 게임을 종료할 수 있습니다. 카드를 뒤집었을 때, 빨간색(하트/다이아몬드) 카드가 나오면 100원을 얻고 까만색(스페이드/클로버) 카드가 나오면 100원을 잃습니다. 최적의 전략으로 행동을 선택한다면 이 게임의 적절한 참가비는 얼마라고 생각하십니까?

Markov Decision Process Approach

- ▶ 현재 남은 (빨간 카드, 검은 카드) 쌍을 상태로 삼는다.
- ▶ Action은 계속 게임을 할지 멈출지 두 가지
- ▶ Transition probability는 남아있는 카드 종류와 숫자에 의해 결정된다.
- ▶ Reward는 게임을 멈추는 순간에만 얻으며 그 전에는 항상 0이다.

Value Iteration

```
import numpy as np

rewards = np.array([np.arange(27) - i for i in range(27)]) * 100
p = lambda i, j : j / (i + j)
V = np.zeros(27 * 27).reshape(27, 27)

for k in range(100000):
    i, j = np.random.randint(0, 27, size=2)
    Q_stop, Q_play = rewards[i, j], 0
    Q_play += (1 - p(i, j)) * V[i - 1, j] if i != 0 else 0
    Q_play += p(i, j) * V[i, j - 1] if j != 0 else 0
    V[i, j] = max(Q_stop, Q_play)

print(V[-1, -1]) # 262.4475548993924
```


Observation #1

Fibonacci

DP 문제를 RL의 관점에서 접근해보면 쉽게 식이 세워지는 경우가 많다.

▶ Fibonacci sequence

- 상태는 n
- No action
- No transition
- Return은 이전 두 상태의 가치의 합

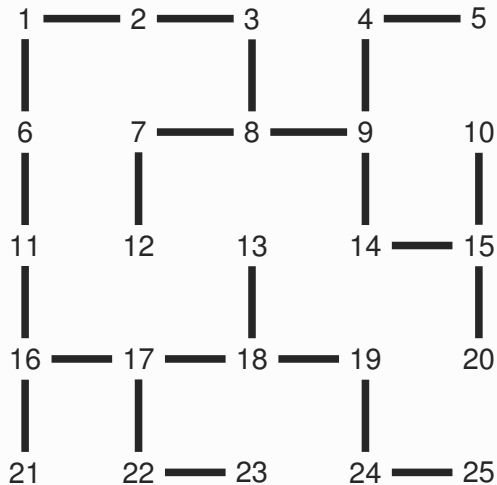
$$V(n) = V(n - 1) + V(n - 2)$$

Observation #1

Viterbi

- ▶ Chain structure를 지닌 graphical model에서 argmax 문제를 푸는 한 가지 방법은 Viterbi 알고리즘
- ▶ HMM에서 길이 T 인 sequence에서 t 까지 어떤 경로를 따라왔든 앞으로의 progression은 t 시점의 상태에 의해서만 결정됨 (상태)
- ▶ No action
- ▶ Transition probability는 미리 정해짐 (혹은 Baum-Welch/EM 알고리즘으로 따로 구함)
- ▶ Reward는 log probability로 결정됨

Message Passing



Observation #2

Message passing과 DP, RL/value iteration 접근의 관계를 고민해보면 재밌는 유사점이 드러난다.

- ▶ Chain 혹은 tree 형태의 graphical model에서는 dynamic programming을 통해 argmax 문제를 정확하게 풀 수 있음
- ▶ Message passing 이라는 이름으로 주로 서술
- ▶ Message passing과 RL/Value iteration의 관계를 고민해볼 수 있음

Message Passing & Value Iteration

- ▶ General graph에서는 approximation을 사용, i.e., loopy belief propagation
 - Loop를 무시하고 마치 tree 인 것처럼 message passing
 - Damping을 통해 convergence를 도움
- ▶ MDP에서 return이 폭발하는 것을 막기 위해 도입하는 방법 중 하나가 discounted reward

Backpropagation

- ▶ Backpropagation도 DP의 일종
- ▶ Backpropagation을 수행하는 그래프는 DAG
- ▶ Message passing 관점에서 바라보면 임의의 노드를 잡고 이전 노드의 gradient 정보를 흘리는 것을 반복하면 언젠가는 수렴하여 정확한 gradient를 계산할 수 있음
 - 물론 매우 비효율적

Therefore psychologically we must keep all the theories in our heads, and every theoretical physicist who is any good knows six or seven different theoretical representations for exactly the same physics.

(Richard Feynman)